**Marathon Match - Solution Description**

**Overview**

Congrats on winning this marathon match. As part of your final submission and in order to receive payment for this marathon match, please complete the following document.

1. **Introduction**

Tell us a bit about yourself, and why you have decided to participate in the contest.

* Name: Roman Solovyev
* Handle: ZF\_Turbo
* Placement you achieved in the MM: 6th place
* About you: I’m working in Institute of Designing Problems in Microelectronics (part of Russian Academy of Sciences) as Leading Researcher. I often take part in machine learning competitions for last 2 years. I have extensive experience with GBM, Neural Nets and Deep Learning as well as with development of CAD programs in different programming languages (C/C++, Python, TCL/TK, PHP etc).
* Why you participated in the MM: I had recent experience with automatic segmentation of satellite images. I wanted to try new neural net architectures for this task based on pre-trained nets for multispectral input.

1. **Solution Development**

How did you solve the problem? What approaches did you try and what choices did you make, and why? Also, what alternative approaches did you consider?

* My main idea was to use new neural net architectures based on pre-trained nets: ResNet50, Inception\_ResNet\_v2, VGG16 etc. These nets were used as encoders. The main problem was that they trained on 3 channel input. I was able to extend first layer to get 5 channel input and recalculate weights. Earlier I mostly trained uninitialized nets based on UNet basic architecture. Switching to other architecture was made because usage of pre-trained nets always gives better result on classification tasks, so I expected it to give better result on segmentation task as well. I tried several neural nets. And encoder Inception\_ResNet\_v2 was the best one for score, but slowest one for speed. ResNet50 encoder gives slightly worse score, but was much faster.
* There are additional 2 channels in DSM and DTM files. They were normalized and added as additional input planes.
* I also find out some problems with masks appeared on black part of images, which influenced training process in negative way, so I added some preprocessing of input. It’s important to look into training data carefully.
* To improve the score Test Time Augmentation technique was added. It requires segmenting the same part of images which were rotated or mirrored. So it’s slow down inference a little, but increase the accuracy.

1. **Final Approach**

Please provide a bulleted description of your final approach. What ideas/decisions/features have been found to be the most important for your solution performance:

* Stage 1 (input preprocessing): Input data was preprocessed and converted from TIFF format to PNG format. Masks were a little bit fixed for empty part of images. The same preprocessing is made for train and test part of data.
* Stage 2 (training of neural net models): To speed up the process all training images are read in memory. To train model random part of images of size 288x288 are taken from all 5 available channels, forming 3D array of shape (5, 288, 288). All this data is given for neural net input to process single forward/backward pass. I use Adam optimizer with relatively small learning rate 0.0001. Loss function is smoothed Dice coefficient: <https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient>. Dice coefficient is slowly growing showing how good model is on current epoch. Training is stopping in case there were no improvements for the last 50 epochs.  
  Actually we train not one but 5 different models, which trained on different subsets of training images. I use here 5Kfold cross-validation method: https://en.wikipedia.org/wiki/Cross-validation\_(statistics)#k-fold\_cross-validation
* Stage 3 (inference of test data): All large images processed with sliding window approach. Window has the size 288x288 and moving along the large image with step 144. We sum up obtained probabilities in the matrix with same size as initial image. Because neural nets have worse result on border, we only use central part with predictions, moving 40 pixels from border of analyzed image. We process image with all 5 models obtained on previous stage. Also Test Time Augmentation technique is used: we process with same model initial image and all possible unique 90-degrees rotations and mirroring and then averaged the probabilities. All of these allow increasing accuracy, but slightly slowdown the model. At the end probability matrix is binarized to 0 or 1 value using threshold = 0.5. Places with 1 value are the buildings. Obtained masks and probabilities are saved as intermediate PNG files.
* Stage 4 (create submission in text format): At this stage we extract connected regions from PNG files with masks obtained on previous stage. There is no complicated post processing. We extract connected regions with area not less than 100 with openCV findContours function and enumerate them.

1. **Open Source Resources, Frameworks and Libraries**

Please specify the name of the open source resource along with a URL to where it’s housed and it’s license type:

* For creating solution I used Python 3.5. Solution is cross platform: <https://www.python.org/downloads/> Or it’s better to use Python bundle like Anaconda: https://www.anaconda.com/download/
* It also required some modules. All of them are free and can be installed with pip or conda binaries from python: keras >=2.0.8, tensorflow >= 1.2.0, numpy >= 1.13, opencv >= 1.1.0, skimage, tifffile

1. **Potential Algorithm Improvements**

Please specify any potential improvements that can be made to the algorithm:

* There are many different pre-trained neural nets which can be used as encoder. Some of them could work better than Inception ResNet v2
* Some papers reported that usage of Dice coefficient as loss functions is worse than usage of mix Binary Cross Entropy and Dice. Also some other experimental loss functions can be tried.
* I believe that some smart post processing related to building separation could increase the score. In current solution close building sometimes marked as one.

1. **Algorithm Limitations**

Please specify any potential limitations with the algorithm:

* Algorithm requires usage of GPU with large amount of memory. >= 11 GB. So it’s Nvidia GTX 1080 Ti or Titan. Its due usage of modern neural net as encoder.

1. **Deployment Guide**

Please provide the exact steps required to build and deploy the code:

1. Step 1 - Unzip archive
2. Step 2 - Put "training" and "testing" folders in "input" folder
3. Step 3 - Build docker: sudo nvidia-docker build -t zfturbo .
4. Step 4 - Run docker: sudo nvidia-docker run -v ~/project:/data -it zfturbo
5. **Final Verification**

Please provide instructions that explain how to train the algorithm and have it execute against sample data:

1. Step 1 - Training example: ./train.sh ./input/training/
2. Step 2 - Testing example: ./test.sh ./input/training/ ./input/testing/ out.txt
3. **Feedback**

Please provide feedback on the following - what worked, and what could have been done better or differently?

* Problem Statement – it was totally fine. I only had small problem with generating text representation of masks with RLE.
* Data – again, ok. I only wish to have more samples available in train set. The great thing is that it had direct data download without Amazon EC2 account.
* Contest – great contest for learning purposes
* Scoring – can’t complain, scoring was fine.

**NOTE**: Please save a copy of this template in word format. Please do not submit a .pdf